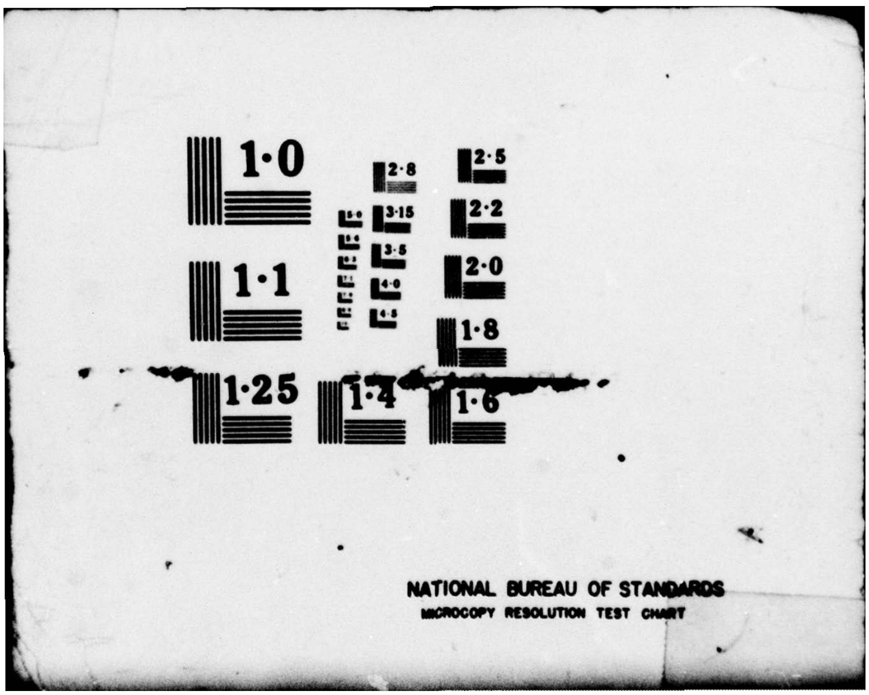
AD-A071 422 STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE F/G 9/4
FIRST ORDER THEORIES OF INDIVIDUAL CONCEPTS AND PROPOSITIONS.(U)
MAR 79 J MCCARTHY MDA903-76-C-0206
UNCLASSIFIED STAN-CS-79-724 NL

1 OF 1
AD
A071422


1·0
2·8
2·5
3·15
2·2
3·6
2·0
4·0
4·5
1·1
1·8
1·25
1·4
1·6
NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

Stanford Artificial Intelligence Laboratory
Memo AIM-325

March 1979

Computer Science Department
Report No. STAN-CS-79-724

FIRST ORDER THEORIES OF INDIVIDUAL CONCEPTS AND PROPOSITIONS

by

John McCarthy

Research sponsored by

Advanced Research Projects Agency
and
National Science Foundation

COMPUTER SCIENCE DEPARTMENT
Stanford University

Stanford Artificial Intelligence Laboratory
Memo AIM-325

March 1979

Computer Science Department
Report No. STAN-CS-79-724

FIRST ORDER THEORIES OF INDIVIDUAL CONCEPTS AND PROPOSITIONS

by

John McCarthy

We discuss first order theories in which *individual concepts* are admitted as mathematical objects along with the things that *reify* them. This allows very straightforward formalizations of knowledge, belief, wanting, and necessity in ordinary first order logic without modal operators. Applications are given in philosophy and in artificial intelligence. This paper will appear in *Machine Intelligence 9*, edited by Donald Michie.

This research was supported by the Advanced Research Projects Agency of the Department of Defense under ARPA Order No. 2494, Contract MDA903-76-C-0206 and National Science Foundation under Contract NSF MCS 78-00524. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Stanford University, or any agency of the U. S. Government.

INTRODUCTION

"*...it seems that hardly anybody proposes to use different variables for propositions and for truth-values, or different variables for individuals and individual concepts.*" - (Carnap 1956, p. 113).

Admitting individual concepts as objects - with concept-valued constants, variables, functions and expressions - allows ordinary first order theories of necessity, knowledge, belief and wanting without modal operators or quotation marks and without the restrictions on substituting equals for equals that either device makes necessary.

According to the ideas of Frege (1892), the meaning of the phrase "*Mike's telephone number*" in the sentence "*Pat knows Mike's telephone number*" is the concept of Mike's telephone number, whereas its meaning in the sentence "*Pat dialed Mike's telephone number*" is the number itself. Thus if we also have "*Mary's telephone number = Mike's telephone number*", then "*Pat dialed Mary's telephone number*" follows, but "*Pat knows Mary's telephone number*" does not.

Frege further proposed that a phrase has a *sense* which is a *concept* and is its *meaning* in *oblique contexts* like knowing and wanting, and a *denotation* which is its *meaning* in *direct contexts*. *Denotations* are the basis of the semantics of first order logic and model theory and are well understood, but *sense* has given more trouble, and the modal treatment of oblique contexts avoids the idea. On the other hand, logicians such as Carnap (1947 and 1956), Church (1951) and Montague (1974) see a need for *concepts* and have proposed formalizations. All these formalizations involve modifying the logic used; ours doesn't modify the logic and is more powerful, because it includes mappings from objects to concepts.

The problem identified by Frege - of suitably limiting the application of the substitutitivity of equals for equals - arises in artificial intelligence as well as in philosophy and linguistics for any system that must represent information about beliefs, knowledge, desires, or logical necessity - regardless of whether the representation is declarative or procedural (as in PLANNER and other AI formalisms).

Our approach involves leaving the logic unchanged and treating concepts as one kind of object in an ordinary first order theory. We shall have one term that denotes Mike's telephone number and a different term denoting the concept of Mike's telephone number instead of having a single term whose denotation is the number and whose sense is a concept of it. The relations among concepts and between concepts and other entities are expressed by formulas of first order logic. Ordinary model theory can then be used to study what spaces of concepts satisfy various sets of axioms.

We treat primarily what Carnap calls *individual concepts* like *Mike's telephone number* or *Pegasus* and not general concepts like *telephone* or *unicorn*. Extension to general concepts seems feasible, but individual concepts provide enough food for thought for the present.

It seems surprising that such a straightforward and easy approach should not have been more fully explored than it apparently has.

KNOWING WHAT AND KNOWING THAT

To assert that Pat knows Mike's telephone number we write

1) *true Know(Pat, Telephone Mike)*

with the following conventions:

1. Parentheses are often omitted for one argument functions and predicates. This purely syntactic convention is not important. Another convention is to capitalize the first letter of a constant, variable or function name when its value is a concept. (We considered also capitalizing the last letter when the arguments are concepts, but it made the formulas ugly).

2. *Mike* is the concept of Mike; i.e. it is the *sense* of the expression "*Mike*". *mike* is Mike himself.

3. *Telephone* is a function that takes a concept of a person into a concept of his telephone number. We will also use *telephone* which takes the person himself into the telephone number itself. Whether the function *Telephone* can be identified with the general concept of a person's telephone number is not settled. For the present, please suppose not.

4. If *P* is a person concept and *X* is another concept, then *Know(P, X)* is an assertion concept or *proposition* meaning that *P knows* the value of *X*. Thus in (1) *Know(Pat, Telephone Mike)* is a proposition and not a truth value. Note that we are formalizing *knowing what* rather than *knowing that* or *knowing how*. For AI and for other practical purposes, *knowing what* seems to be the most useful notion of the three. In English, *knowing what* is written *knowing whether* when the "knowand" is a proposition.

5. *true Q* is the truth value, *t* or *f*, of the proposition *Q*, and we must write *true Q* in order to assert *Q*. Later we will consider formalisms in which *true* has a another argument - a *situation*, a *story*, a *possible world*, or even a *partial possible world* (a notion we suspect will eventually be found necessary).

6. The formulas are in a sorted first order logic with functions and equality. Knowledge, necessity, etc. will be discussed without extending the logic in any way - solely by the introduction of predicate and function symbols subject to suitable axioms. In the present informal treatement, we will not be explicit about sorts, but we will use different letters for variables of different sorts.

The reader may be nervous about what is meant by *concept*. He will have to remain nervous; no final commitment will be made in this paper. The formalism is compatible with many possibilities, and these can be compared using the models of their first order theories. Actually, this paper isn't much motivated by the philosophical question of what concepts really are. The goal is more to make a formal structure that can be used to represent facts about knowledge and belief so that a computer program can reason about who has what knowledge in order to solve problems. From either the philosophical or the AI point of view, however, if (1) is to be reasonable, it must not follow from (1) and the fact that Mary's telephone number is the same as Mike's, that Pat knows Mary's telephone number.

The proposition that Joe knows *whether* Pat knows Mike's telephone number, is written

2) $Know(Joe, Know(Pat, Telephone\ Mike))$,

and asserting it requires writing

3) $true\ Know(Joe, Know(Pat, Telephone\ Mike))$,

while the proposition that Joe knows *that* Pat knows Mike's telephone number is written

4) $K(Joe, Know(Pat, Telephone\ Mike))$,

where $K(P,Q)$ is the proposition that P knows *that* Q. English does not treat knowing a proposition and knowing an individual concept uniformly; knowing an individual concept means knowing its value while knowing a proposition means knowing that it has a particular value, namely t. There is no reason to impose this infirmity on robots.

We first consider systems in which corresponding to each concept X, there is a thing x of which X is a concept. Then there is a function *denot* such that

5) $x = denot\ X$.

Functions like *Telephone* are then related to *denot* by equations like

6) $\forall P1\ P2.(denot\ P1 = denot\ P2 \supset denot\ Telephone\ P1 = denot\ Telephone\ P2)$.

We call *denot* X the *denotation* of the concept X, and (6) asserts that the denotation of the concept of P's telephone number depends only on the denotation of the concept P. The variables in (6) range over concepts of persons, and we regard (6) as asserting that *Telephone is extensional* with respect to *denot*. Note that our *denot* operates on concepts rather than on expressions; a theory of expressions will also need a denotation function. From (6) and suitable logical axioms follows the existence of a function *telephone* satisfying

7) $\forall P.(denot\ Telephone\ P = telephone\ denot\ P)$.

Know is extensional with respect to *denot* in its first argument, and this is expressed by

8) $\forall P1\ P2\ X.(denot\ P1 = denot\ P2 \supset denot\ Know(P1, X) = denot\ Know(P2, X))$,

but it is not extensional in its second argument. We can therefore define a predicate $know(p, X)$ satisfying

9) $\forall P\ X.(true\ Know(P, X) \equiv know(denot\ P, X))$.

(Note that all these predicates and functions are entirely extensional in the underlying logic, and the notion of extensionality presented here is relative to *denot*.)

The predicate *true* and the function *denot* are related by

10) $\forall Q.(true\ Q \equiv (denot\ Q = t))$

provided truth values are in the range of *denot*, and *denot* could also be provided with a (*partial*) *possible world* argument.

When we don't assume that all concepts have denotations, we use a predicate $denotes(X,x)$ instead of a function. The extensionality of *Telephone* would then be written

11) $\forall P1\ P2\ x\ u.(denotes(P1,x) \wedge denotes(P2,x) \wedge denotes(Telephone\ P1,u) \supset denotes(Telephone\ P2,u))$.

We now introduce the function *Exists* satisfying

12) $\forall X.(true\ Exists\ X \equiv \exists x.denotes(X,x))$.

Suppose we want to assert that Pegasus is a horse without asserting that Pegasus exists. We can do this by introducing the predicate *Ishorse* and writing

13) *true Ishorse Pegasus*

which is related to the predicate *ishorse* by

14) $\forall X\ x.(denotes(X,x) \supset (ishorse\ x \equiv true\ Ishorse\ X))$.

In this way, we assert extensionality without assuming that all concepts have denotations. *Exists* is extensional in this sense, but the corresponding predicate *exists* is identically true and therefore dispensable.

In order to combine concepts propositionally, we need analogs of the propositional operators such as *And*, which we shall write as an infix and axiomatize by

15) $\forall Q1\ Q2.(true(Q1\ And\ Q2) \equiv true\ Q1 \wedge true\ Q2)$.

The corresponding formulas for *Or*, *Not*, *Implies*, and *Equiv* are

16) $\forall Q1\ Q2.(true(Q1\ Or\ Q2) \equiv true\ Q1 \vee true\ Q2)$,

17) $\forall Q.(true(Not\ Q) \equiv \neg\ true\ Q)$,

18) $\forall Q1\ Q2.(true(Q1\ Implies\ Q2) \equiv true\ Q1 \supset true\ Q2)$,

and

19) $\forall Q1\ Q2.(true(Q1\ Equiv\ Q2) \equiv (true\ Q1 \equiv true\ Q2))$.

The equality symbol "=" is part of the logic so that $X = Y$ asserts that X and Y are the same concept. To write propositions expressing equality, we introduce $Equal(X,Y)$ which is a proposition that X and Y denote the same thing if anything. We shall want axioms

20) $\forall X.true\ Equal(X,X)$,

21) $\forall X\ Y.(true\ Equal(X,Y) \equiv true\ Equal(Y,X))$,

and

22) $\forall X\ Y\ Z.(true\ Equal(X,Y) \wedge true\ Equal(Y,Z) \supset true\ Equal(X,Z)$

making *true Equal*(X,Y) an equivalence relation, and

23) $\forall X\ Y\ x.(true\ Equal(X,Y) \wedge denotes(X,x) \supset denotes(Y,x))$

which relates it to equality in the logic. We can make the concept of equality *essentially* symmetric by replacing (21) by

24) $\forall X\ Y.Equal(X,Y) = Equal(Y,X)$,

i.e. making the two expressions denote the *same concept*.

The statement that Mary has the same telephone as Mike is asserted by

25) *true Equal(Telephone Mary, Telephone Mike),*

and it obviously doesn't follow from this and (1) that

26) *true Know(Pat, Telephone Mary).*

To draw this conclusion we need something like

27) *true K(Pat, Equal(Telephone Mary, Telephone Mike))*

and suitable axioms about knowledge.

If we were to adopt the convention that a proposition appearing at the outer level of a sentence is asserted and were to regard the denotation-valued function as standing for the sense-valued function when it appears as the second argument of *Know*, we would have a notation that resembles ordinary language in handling obliquity entirely by context. There is no guarantee that general statements could be expressed unambiguously without circumlocution; the fact that the principles of intensional reasoning haven't yet been stated is evidence against the suitability of ordinary language for stating them.

FUNCTIONS FROM THINGS TO CONCEPTS OF THEM

While the relation *denotes*(X,x) between concepts and things is many-one, functions going from things to certain concepts of them seem useful. Some things such as numbers can be regarded as having *standard* concepts. Suppose that *Concept1* n gives a standard concept of the number n, so that

28) $\forall n.(denot\ Concept1\ n = n)$.

We can then have simultaneously

29) *true Not Knew(Kepler, Number Planets)*

and

30) *true Knew(Kepler, Composite Concept1 denot Number Planets).*

(We have used *Knew* instead of *Know*, because we are not now concerned with formalizing tense.)

(30) can be condensed using *Composite1* which takes

a number into the proposition that it is composite, i.e.

31) *Composite1 n = Composite Concept1 n*

getting

32) *true Knew(Kepler, Composite1 denot Number Planets).*

A further condensation can be achieved using *Composite2* defined by

33) *Composite2 N = Composite Concept1 denot N,*

letting us write

34) *true Knew(Kepler, Composite2 Number Planets),*

which is true even though

35) *true Knew(Kepler, Composite Number Planets)*

is false. (35) is our formal expression of "*Kepler knew that the number of planets is composite*", while (30), (32), and (34) each expresses a proposition that can only be stated awkwardly in English, e.g. as "*Kepler knew that a certain number is composite, where this number (perhaps unbeknownst to Kepler) is the number of planets*".

We may also want a map from things to concepts of them in order to formalize a sentence like, "*Lassie knows the location of all her puppies*". We write this

36) $\forall x.(ispuppy(x, lassie) \supset true\ Knowd(Lassie, Locationd\ Conceptd\ x))$.

Here *Conceptd* takes a puppy into a dog's concept of it, and *Locationd* takes a dog's concept of a puppy into a dog's concept of its location. The axioms satisfied by *Knowd*, *Locationd* and *Conceptd* can be tailored to our ideas of what dogs know.

A suitable collection of functions from things to concepts might permit a language that omitted some individual concepts like *Mike* (replacing it with *Conceptx mike*) and wrote many sentences with quantifiers over things rather than over concepts. However, it is still premature to apply Occam's razor. It may be possible to avoid concepts as objects in expressing particular facts but impossible to avoid them in stating general principles.

RELATIONS BETWEEN KNOWING WHAT AND KNOWING THAT

As mentioned before, "*Pat knows Mike's telephone number*" is written

37) *true Know(Pat, Telephone Mike)*.

We can write "*Pat knows Mike's telephone number is 333-3333*"

38) *true K(Pat, Equal(Telephone Mike, Concept1 "333-3333")*

where $K(P, Q)$ is the proposition that *denot(P)* knows the proposition Q and *Concept1*("333-3333") is some standard concept of that telephone number.

The two ways of expressing knowledge are somewhat interdefinable, since we can write

39) $K(P, Q) = (Q \ And \ Know(P, Q))$,

and

40) $true \ Know(P, X) \equiv \exists A.(constant \ A \wedge true \ K(P, Equal(X, A)))$.

Here *constant A* asserts that A is a constant, i.e. a concept such that we are willing to say that P knows X if he knows it equals A. This is clear enough for some domains like integers, but it is not obvious how to treat knowing a person.

Using the *standard concept* function *Concept1*, we might replace (40) by

41) $true \ Know(P, X) \equiv \exists a.true \ K(P, Equal(X, Concept1 \ a))$

with similar meaning.

(40) and (41) expresses a *denotational* definition of *Know* in terms of K. A *conceptual* definition seems to require something like

42) $\forall P \ X.(Know(P, X) = Exists \ X \ And \ K(P, Equal(X, Concept2 \ denot \ X)))$,

where *Concept2* is a suitable function from things to concepts and may not be available for all sorts of objects.

UNQUANTIFIED MODAL LOGIC

In *unquantified modal logic*, the arguments of the modal functions will not involve quantification although quantification occurs in the outer logic.

Nec Q is the proposition that the proposition Q is necessary, and *Poss Q* is the proposition that it is possible. To assert necessity or possibility we must write *true Nec Q* or *true Poss Q*. This can be abbreviated by defining $nec \ Q \equiv true \ Nec \ Q$ and *poss Q* correspondingly. However, since *nec* is a predicate in the logic with t and f as values, *nec Q* cannot be an argument of *nec* or *Nec*.

Before we even get to modal logic proper we have a decision to make - shall *Not Not Q* be considered the same proposition as Q, or is it merely extensionally equivalent? The first is written

43) $\forall Q.\ Not\ Not\ Q = Q,$

and the second

44) $\forall Q.true\ Not\ Not\ Q \equiv true\ Q.$

The second follows from the first by substitution of equals for equals.

In *Meaning and Necessity*, Carnap takes what amounts to the first alternative, regarding concepts as L-equivalence classes of expressions. This works nicely for discussing necessity, but when he wants to discuss knowledge without assuming that everyone knows Fermat's last theorem if it is true, he introduces the notion of *intensional isomorphism* and has knowledge operate on the equivalence classes of this relation.

If we choose the first alternative, then we may go on to identify any two propositions that can be transformed into each other by Boolean identities. This can be assured by a small collection of propositional identities like (43) including associative and distributive laws for conjunction and disjunction, De Morgan's law, and the laws governing the propositions T and F. In the second alternative we will want the extensional forms of the same laws. When we get to quantification a similar choice will arise, but if we choose the first alternative, it will be undecideable whether two expressions denote the same concept. I doubt that considerations of linguistic usage or usefulness in AI will unequivocally recommend one alternative, so both will have to be studied.

Actually there are more than two alternatives. Let M be the free algebra built up from the "atomic" concepts by the concept forming function symbols. If $==$ is an equivalence relation on M such that

45) $\forall X1\ X2 \in M.((X1 == X2) \supset (true\ X1 \equiv true\ X2)),$

then the set of equivalence classes under $==$ may be taken as the set of concepts.

Similar possibilities arise in modal logic. We can choose between the *conceptual identity*

46) $\forall Q.(Poss\ Q = Not\ Nec\ Not\ Q),$

and the weaker extensional axiom

47) $\forall Q.(true\ Poss\ Q \equiv true\ Not\ Nec\ Not\ Q).$

We will write the rest of our modal axioms in extensional form.

We have

48) $\forall Q.(true\ Nec\ Q \supset true\ Q),$

and

49) $\forall Q1\ Q2.(true\ Nec\ Q1 \wedge true\ Nec(Q1\ Implies\ Q2) \supset true\ Nec\ Q2)$.

yielding a system equivalent to von Wright's T.

S4 is given by

50) $\forall Q.(true\ Nec\ Q \equiv true\ Nec\ Nec\ Q)$,

and S5 by

51) $\forall Q.(true\ Poss\ Q \equiv true\ Nec\ Poss\ Q)$.

Actually, there may be no need to commit ourselves to a particular modal system. We can simultaneously have the functions *NecT*, *Nec4* and *Nec5*, related by axioms such as

52) $\forall Q.(true\ Nec4\ Q \supset true\ Nec5\ Q)$

which would seem plausible if we regard S4 as corresponding to provability in some system and S5 as truth in the intended model of the system.

Presumably we shall want to relate necessity and equality by the axiom

53) $\forall X.true\ Nec\ Equal(X, X)$.

Certain of Carnap's proposals translate to the stronger relation

54) $\forall X\ Y.(X = Y \equiv true\ Nec\ Equal(X, Y))$

which asserts that two concepts are the same if and only if the equality of what they may denote is necessary.

MORE PHILOSOPHICAL EXAMPLES - MOSTLY WELL KNOWN

Some sentences that recur as examples in the philosophical literature will be expressed in our notation so the treatments can be compared.

First we have "*The number of planets* = 9" and "*Necessarily* 9 = 9" from which one doesn't want to deduce "*Necessarily the number of planets* = 9". This example is discussed by Quine (1961) and (Kaplan 1969). Consider the sentences

55) $\neg nec\ Equal(Number\ Planets,\ Concept1\ 9)$

and

56) $nec\ Equal(Concept1\ number\ planets, Concept1\ 9)$.

Both are true. (55) asserts that it is not necessary that the number of planets be 9, and (56) asserts that the number of planets, once determined, is a number that is necessarily equal to 9. It is a major virtue of our formalism that both meanings can be expressed and are readily distinguished. Sustitutivity of equals nolds in the logic but causes no trouble, because "*The number of planets* = 9" may be written

57) *number(planets)* = 9

or, using concepts, as

58) *true Equal(Number Planets, Concept*1 9),

and "*Necessarily* 9=9" is written

59) *nec Equal(Concept*1 9, *Concept*1 9),

and these don't yield the unwanted conclusion.

Ryle used the sentences "*Baldwin is a statesman*" and "*Pickwick is a fiction*" to illustrate that parallel sentence construction does not always give parallel sense. The first can be rendered in four ways, namely *true Statesman Baldwin* or *statesman denot Baldwin* or *statesman baldwin* or *statesman*1 *Baldwin* where the last asserts that the concept of Baldwin is one of a statesman. The second can be rendered only as as *true Fiction Pickwick* or *fiction*1 *Pickwick*.

Quine (1961) considers illegitimate the sentence

60) $(\exists x)$(*Philip is unaware that x denounced Catiline*)

obtained from "*Philip is unaware that Tully denounced Catiline*" by existential generalization. In the example, we are also supposing the truth of *Philip is aware that Cicero denounced Catiline*". These sentences are related to (perhaps even explicated by) several sentences in our system. *Tully* and *Cicero* are taken as distinct concepts. The person is called *tully* or *cicero* in our language, and we have

61) *tully* = *cicero*,

62) *denot Tully* = *cicero*

and

63) *denot Cicero* = *cicero*.

We can discuss Philip's concept of the person Tully by introducing a function *Concept*2(p1, p2) giving for some persons p1 and p2, p1's concept of p2. Such a function need not be unique or always defined, but in the present case, some of our information may be conveniently expressed by

64) *Concept*2(*philip*, *tully*) = *Cicero*,

asserting that Philip's concept of the person Cicero is *Cicero*. The basic assumptions of Quine's example also include

65) $true\ K(Philip, Denounced(Cicero, Catiline))$

and

66) $\neg true\ K(Philip, Denounced(Tully, Catiline))$,

From (62), ... ,(66) we can deduce

67) $\exists P.true\ Denounced(P, Catiline)\ And\ Not\ K(Philip, Denounced(P, Catiline))$,

from (66) and others, and

68) $\neg\exists p.(denounced(p, catiline) \wedge \neg true\ K(Philip, Denounced(Concept2(philip, p),\ Catiline)))$

using the additional hypotheses

69) $\forall p.(denounced(p, catiline) \supset p = cicero)$,

70) $denot\ Catiline = catiline$,

and

71) $\forall P1\ P2.(denot\ Denounced(P1, P2) = denounced(denot\ P1, denot\ P2))$.

Presumably (67) is always true, because we can always construct a concept whose denotation is Cicero unbeknownst to Philip. The truth of (68) depends on Philip's knowing that someone denounced Catiline and on the map $Concept2(p1, p2)$ that gives one person's concept of another. If we refrain from using a silly map that gives something like *Denouncer(Catiline)* as its value, we can get results that correspond to intuition.

The following sentence attributed to Russell is is discussed by Kaplan: "*I thought that your yacht was longer than it is*". We can write it

72) $true\ Believed(I, Greater(Length\ Youryacht, Concept1\ denot\ Length\ Youryacht))$

where we are not analyzing the pronouns or the tense, but are using *denot* to get the actual length of the yacht and *Concept1* to get back a concept of this true length so as to end up with a proposition that the length of the yacht is greater than that number. This looks problematical, but if it is consistent, it is probably useful.

In order to express "*Your yacht is longer than Peter thinks it is.*", we need the expression $Denot(Peter, X)$ giving a concept of what Peter thinks the value of X is. We now write

73) $longer(youryacht, denot\ Denot(Peter, Length\ Youryacht))$,

but I am not certain this is a correct translation.

Quine (1956) discusses an example in which Ralph sees Bernard J. Ortcutt skulking about and concludes that he is a spy, and also sees him on the beach, but doesn't recognize him as the same person. The facts can be expresed in our formalism by equations

74) *true Believe(Ralph, Isspy P1)*

and

75) *true Believe(Ralph, Not Isspy P2)*

where *P*1 and *P*2 are concepts satisfying

*denot P*1 = *ortcutt* and *denot P*2 = *ortcutt*. *P*1 and *P*2 are further described by sentences relating them to the circumstances under which Ralph formed them.

We can still consider a simple sentence involving the persons as things – write it *believespy(ralph, ortcutt)*, where we define

76) $\forall p1\ p2.(believespy(p1, p2) \equiv true\ Believe(Concept1\ p1, Isspy\ Concept7\ p2)$

using suitable mappings *Concept*1 and *Concept*7 from persons to concepts of persons. We might also choose to define *believespy* in such a way that it requires *true Believe(Concept*1 *p*1, *Isspy P)* for several concepts *P* of *p*2, e.g. the concepts arising from all *p*1's encounters with *p*2 or his name. In this case *believespy(ralph, ortcutt)* will be false and so would a corresponding *notbelievespy(ralph, ortcutt)*. However, the simple-minded predicate *believespy*, suitably defined, may be quite useful for expressing the facts necessary to predict someone's behavior in simpler circumstances.

Regarded as an attempt to explicate the sentence "*Ralph believes Ortcutt is a spy*", the above may be considered rather tenuous. However, we are proposing it as a notation for expressing Ralph's beliefs about Ortcutt so that correct conclusions may be drawn about Ralph's future actions. For this it seems to be adequate.

PROPOSITIONS EXPRESSING QUANTIFICATION

As the examples of the previous sections have shown, admitting concepts as objects and introducing standard concept functions makes "quantifying in" rather easy. However, forming propositions and individual concepts by quantification requires new ideas and additional formalism.

We want to continue describing concepts within first order logic with no logical extensions. Therefore, in order to form new concepts by quantification and description, we introduce functions *All*, *Exist*, and *The* such that *All(V, P)* is (approximately) the proposition that *for all values of V P is true*, *Exist(V, P)* is the corresponding existential proposition, and *The(V, P)* is the concept of *the V such that P*.

Since *All* is to be a function, *V* and *P* must be objects in the logic. However, *V* is semantically a variable in the formation of *All(V, P)*, *etc.*, and we will call such objects *inner*

variables so as to distinguish them from variables in the logic. We will use V, sometimes with subscripts, for a logical variable ranging over inner variables. We also need some constant symbols for inner variables (got that?), and we will use doubled letters, sometimes with subscripts, for these. XX will be used for individual concepts, PP for persons, and QQ for propositions.

The second argument of *All* and friends is a "proposition with variables in it", but remember that these variables are inner variables which are constants in the logic. Got that? We won't introduce a special term for them, but will generally allow concepts to include inner variables. Thus concepts now include inner variables like XX and PP, and concept forming functions like *Telephone* and *Know* take the generalized concepts as arguments.

Thus

77) *Child(Mike, PP) Implies Equal(Telephone PP, Telephone Mike)*

is a proposition with the inner variable PP in it to the effect that if PP is a child of Mike, then his telephone number is the same as Mike's, and

78) *All(PP, Child(Mike, PP) Implies Equal(Telephone PP, Telephone Mike))*

is the proposition that all Mike's children have the same telephone number as Mike. Existential propositions are formed similarly to universal ones, but the function *Exist* introduced here should not be confused with the function *Exists* applied to individual concepts introduced earlier.

In forming individual concepts by the description function *The*, it doesn't matter whether the object described exists. Thus

79) *The(PP, Child(Mike, PP))*

is the concept of Mike's only child. *Exists The(PP, Child(Mike, PP))* is the proposition that the described child exists. We have

80) *true Exists The(PP, Child(Mike, PP))* ≡ *true(Exist(PP, Child(Mike, PP) And All(PP1, Child(Mike, PP1) Implies Equal(PP, PP1))))*,

but we may want the equality of the two propositions, i.e.

81) *Exists The(PP, Child(Mike, PP))* = *Exist(PP, Child(Mike, PP) And All(PP1, Child(Mike, PP1) Implies Equal(PP, PP1)))*.

This is part of general problem of when two logically equivalent concepts are to be regarded as the same.

In order to discuss the truth of propositions and the denotation of descriptions, we introduce *possible worlds* reluctantly and with an important difference from the usual treatment. We need them to give values to the inner variables, and we can also use them for axiomatizing the modal operators, knowledge, belief and tense. However, for axiomatizing quantification, we also need a function α such that

82) $\pi' = \alpha(V, x, \pi)$

is the possible world that is the same as the world π except that the inner variable V has the value x instead of the value it has in π. In this respect our possible worlds resemble the *state vectors* or *environments* of computer science more than the possible worlds of the Kripke treatment of modal logic. This Cartesian product structure on the space of possible worlds can also be used to treat counterfactual conditional sentences.

Let $\pi 0$ be the actual world. Let $true(P, \pi)$ mean that the proposition P is true in the possible world π. Then

83) $\forall P.(true\ P \equiv true(P, \pi 0))$.

Let $denotes(X, x, \pi)$ mean that X denotes x in π, and let $denot(X, \pi)$ mean the denotation of X in π when that is defined.

The truth condition for $All(V, P)$ is then given by

84) $\forall \pi\ V\ P.(true(All(V, P), \pi) \equiv \forall x.true(P, \alpha(V, x, \pi))$.

Here V ranges over inner variables, P ranges over propositions, and x ranges over things. There seems to be no harm in making the domain of x depend on π. Similarly

85) $\forall \pi\ V\ P.(true(Exist(V, P), \pi) \equiv \exists x.true(P, \alpha(V, x, \pi))$.

The meaning of $The(V, P)$ is given by

86) $\forall \pi\ V\ P\ x.(true(P, \alpha(V, x, \pi)) \wedge \forall y.(true(P, \alpha(V, y, \pi)) \supset y = x) \supset denotes(The(V, P), x, \pi))$

and

87) $\forall \pi\ V\ P.(\neg \exists ! x.true(P, \alpha(V, x, \pi)) \supset \neg true\ Exists\ The(V, P))$.

We also have the following "syntactic" rules governing propositions involving quantification:

88) $\forall \pi\ Q1\ Q2\ V.(absent(V, Q1) \wedge true(All(V, Q1\ Implies\ Q2), \pi) \supset true(Q1\ Implies\ All(V, Q2), \pi))$

and

89) $\forall \pi\ V\ Q\ X.(true(All(V, Q), \pi) \supset true(Subst(X, V, Q), \pi))$.

where $absent(V, X)$ means that the variable V is not present in the concept X, and $Subst(X, V, Y)$ is the concept that results from substituting the concept X for the variable V in the concept Y. *absent* and *Subst* are characterized by the following axioms:

90) $\forall V1\ V2.(absent(V1, V2) \equiv V1 \neq V2)$,

91) $\forall V\ P\ X.(absent(V, Know(P, X)) \equiv absent(V, P) \wedge absent(V, X))$,

axioms similar to (91) for other conceptual functions,

92) $\forall V\ Q.absent(V, All(V, Q))$,

93) $\forall V\ Q.absent(V, Exist(V, Q))$,

94) $\forall V\ Q.absent(V, The(V, Q))$,

95) $\forall V\ X.Subst(V, V, X) = X$,

96) $\forall X\ V.Subst(X, V, V) = X$,

97) $\forall X\ V\ P\ Y.(Subst(X, V, Know(P, Y)) = Know(Subst(X, V, P), Subst(X, V, Y)))$,

axioms similar to (97) for other functions,

98) $\forall X\ V\ Q.(absent(V, Y) \supset Subst(X, V, Y) = Y)$,

99) $\forall X\ V1\ V2\ Q.(V1 \neq V2 \wedge absent(V2, X) \supset Subst(X, V1, All(V2, Q)) = All(V2, Subst(X, V1, Q)))$,

and corresponding axioms to (99) for *Exist* and *The*.

Along with these comes the axiom that binding kills variables, i.e.

100) $\forall V1\ V2\ Q.(All(V1, Q) = All(V2, Subst(V2, V1, Q)))$.

The functions *absent* and *Subst* play a "syntactic" role in describing the rules of reasoning and don't appear in the concepts themselves. It seems likely that this is harmless until we want to form concepts of the laws of reasoning.

We used the Greek letter π for possible worlds, because we did not want to consider a possible world as a thing and introduce concepts of possible worlds. Reasoning about reasoning may require such concepts or else a formulation that doesn't use possible worlds.

Martin Davis (in conversation) pointed out the advantages of an alternate treatment avoiding possible worlds in case there is a single domain of individuals each of which has a standard concept. Then we can write

101) $\forall V\ Q.(true\ All(V, Q) \equiv \forall x.true\ Subst(Concept1\ x, V, Q))$.

POSSIBLE APPLICATIONS TO ARTIFICIAL INTELLIGENCE

The foregoing discussion of concepts has been mainly concerned with how to translate into a suitable formal language certain sentences of ordinary language. The success of the formalization is measured by the extent to which the logical consequences of these sentences in the formal system agree with our intuitions of what these consequences should be. Another goal of

the formalization is to develop an idea of what concepts really are, but the possible formalizations have not been explored enough to draw even tentative conclusions about that.

For artificial intelligence, the study of concepts has yet a different motivation. Our success in making computer programs with *general intelligence* has been extremely limited, and one source of the limitation is our inability to formalize what the world is like in general. We can try to separate the problem of describing the general aspects of the world from the problem of using such a description and the facts of a situation to discover a strategy for achieving a goal. This is called separating the *epistemological* and the *heuristic* parts of the artificial intelligence problem and is discussed in (McCarthy and Hayes 1969).

We see the following potential uses for facts about knowledge:

1. A computer program that wants to telephone someone must reason about who knows the number. More generally, it must reason about what actions will obtain needed knowledge. Knowledge in books and computer files must be treated in a parallel way to knowledge held by persons.

2. A program must often determine that it does not know something or that someone else doesn't. This has been neglected in the usual formalizations of knowledge, and methods of proving possibility have been neglected in modal logic. Christopher Goad (to be published) has shown how to prove ignorance by proving the existence of possible worlds in which the sentence to be proved unknown is false. Presumably proving one's own ignorance is a stimulus to looking outside for the information. In competitive situations, it may be important to show that a certain course of action will leave competitors ignorant.

3. Prediction of the behavior of others depends on determining what they believe and what they want.

It seems to me that AI applications will especially benefit from first order formalisms of the kind described above. First, many of the present problem solvers are based on first order logic. Morgan (1976) in discussing theorem proving in modal logic also translates modal logic into first order logic. Second, our formalisms leaves the syntax and semantics of statements not involving concepts entirely unchanged, so that if knowledge or wanting is only a small part of a problem, its presence doesn't affect the formalization of the other parts.

In Appendix I, we give a set of axioms for knowledge that permits deduction from "*Pat knows Mike's telephone number*" and *Pat wants Joe to know Mike's telephone number*" that *Joe will know Mike's telephone number*". Treatments of the "dynamics" of knowledge are a first step towards AI applications. The axiomatization is *quasi-static*, i.e. each action takes a situation into a definite resulting situation, and there are no concurrent processes.

The special premisses are written *true(world, Want(Pat, Know(Joe, Telephone Mike)))* and *true(world, Know(Pat, Telephone Mike))*, and the conclusion is *true(world, Future Know(Joe, Telephone Mike))*.

The proof from these axioms that Joe will know Mike's telephone number has about 15 steps. Since there is only one action - Pat telling Joe Mike's telephone number, the frame problem (McCarthy and Hayes 1969) doesn't arise. A more elaborate example in which Joe wants to know

Mike's telephone number, tells Pat that fact, and leading to Pat telling Joe the number has been partially worked out. but the treatment is not very satisfactory. Several frame axioms are required, the proof would be quite long, and the previous result cannot be used as a lemma because its statement doesn't say what remains unchanged when Pat tells Joe Mike's number.

Even the fifteen step proof doesn't model human reasoning, or the way computer programs should be designed to reason. Namely, the particular result is obtained by substitution from a general statement about what to do when a person or machine wants another to know a fact. Therefore, there is no reason to deduce it each time it is needed. Moreover, as the M.I.T. AI school has emphasized, this general fact should be stored so as to be triggered specifically by the desire that another person shall know something.

ABSTRACT LANGUAGES

The way we have treated concepts in this paper, especially when we put variables in them, suggests trying to identify them with terms in some language. It seems to me that this can be done provided we use a suitable notion of *abstract language*.

Ordinarily a language is identified with a set of strings of symbols taken from some alphabet. McCarthy (1963) introduces the idea of *abstract syntax*, the idea being that it doesn't matter whether sums are represented $a+b$ or $+ab$ or $ab+$ or by the integer 2^a3^b or by the LISP S-expression (PLUS A B), so long as there are predicates for deciding whether an expression is a sum and functions for forming sums from summands and functions for extracting the summands from the sum. In particular, abstract syntax facilitates defining the semantics of programming languages, and proving the properties of interpreters and compilers. From that point of view, one can refrain from specifying any concrete representation of the "expressions" of the language and consider it merely a collection of abstract synthetic and analytic functions and predicates for forming, discriminating and taking apart *abstract expressions*. However, the languages considered at that time always admitted representations as strings of symbols.

If we consider concepts as a free algebra on basic concepts, then we can regard them as strings of symbols on some alphabet if we want to, assuming that we don't object to a non-denumerable alphabet or infinitely long expressions if we want standard concepts for all the real numbers. However, if we want to regard $Equal(X, Y)$ and $Equal(Y, X)$ as the same concept, and hence as the same "expression" in our language, and we want to regard expressions related by renaming bound variables as denoting the same concept, then the algebra is no longer free, and regarding concepts as strings of symbols becomes awkward even if possible.

It seems better to accept the notion of *abstract language* defined by the collection of functions and predicates that form, discriminate, and extract the parts of its "expressions". In that case it would seem that concepts can be identified with expressions in an abstract language.

BIBLIOGRAPHY

The treatment given here should be compared with that in (Church 1951b) and in (Morgan 1976). Church introduces what might be called a two-dimensional type structure. One dimension permits higher order functions and predicates as in the usual higher order logics. The second dimension permits concepts of concepts, etc. No examples or applications are given. It seems to me that concepts of concepts will be eventually required, but this can still be done within first order logic.

Morgan's motivation is to use first order logic theorem proving programs to treat modal logic. He gives two approaches. The syntactic approach - which he applies only to systems without quantifiers - uses operations like our *And* to form compound propositions from elementary ones. Provability is then axiomatized in the outer logic. His semantic approach uses axiomatizations of the Kripke accessibility relation between possible worlds. It seems to me that our treatment can be used to combine both of Morgan's methods, and has two further advantages. First, concepts and individuals can be separately quantified. Second, functions from things to concepts of them permit relations between concepts of things that could not otherwise be expressed.

Although the formalism leads in almost the opposite direction, the present paper is much in the spirit of (Carnap 1956). We appeal to his ontological tolerance in introducing concepts as objects, and his section on intensions for robots expresses just the attitude required for artificial intelligence applications.

We have not yet investigated the matter, but plausible axioms for necessity or knowledge expressed in terms of concepts may lead to the paradoxes discussed in (Montague and Kaplan 1961) and (Montague 1963). Our intuition is that the paradoxes can be avoided by restricting the axioms concerning knowledge of facts about knowledge and necessity of statements about necessity. The restrictions will be somewhat unintuitive as are the restrictions necessary to avoid the paradoxes of naive set theory.

Chee K. Yap (1977) proposes *Virtual Semantics* for intensional logics as a generalization of Carnap's individual concepts. Apart from the fact that Yap does not stay within conventional first order logic, we don't yet know the relation between his work and that described here.

REFERENCES:

Carnap, Rudolf (1956), *Meaning and Necessity*, University of Chicago Press.

Church, Alonzo (1951a), The Need for Abstract Entities in Semantic Analysis, in *Contributions to the Analysis and Synthesis of Knowledge*, Proceedings of the American Academy of Arts and Sciences, 80, No. 1 (July 1951), 100-112. Reprinted in *The Structure of Language*, edited by Jerry A. Fodor and Jerrold Katz, Prentice-Hall 1964

(1951b), A formulation of the logic of sense and denotation. In: P. Henle (ed.), *Essays in honor of Henry Sheffer*, pp. 3-24. New York.

Frege, Gottlob (1892), Uber Sinn und Bedeutung. *Zeitschrift fur Philosophie und Philosophische Kritik* 100:25-50. Translated by H. Feigl under the title "On Sense and Nominatum" in H. Feigl and W. Sellars (eds.) *Readings in Philosophical Analysis*, New York 1949. Translated by M. Black under the title "On Sense and Reference" in P. Geach and M. Black, *Translations from the Philosophical Writings of Gottlob Frege*, Oxford, 1952.

Kaplan, David (1969), Quantifying In, from *Words and Objections: Essays on the Work of W.V. Quine*, edited by D. Davidson and J. Hintikka, (Dordrecht-Holland: D. Reidel Publishing Co.), pp. 178-214. Reprinted in (Linsky 1971).

Kaplan, David and Montague, Richard (1960), A Paradox Regained, *Notre Dame Journal of Formal Logic* 1:79-90. Reprinted in (Montague 1974).

Linsky, Leonard, ed.(1971) *Reference and Modality*, Oxford Readings in Philosophy, Oxford University Press.

McCarthy, J. (1963), Towards a Mathematical Science of Computation, in *Proceedings of IFIP Congress* 1962, North-Holland Publishing Co., Amsterdam.

McCarthy, J. and Hayes, P.J. (1969) Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence* 4, pp. 463-502 (eds Meltzer, B. and Michie, D.). Edinburgh: Edinburgh University Press.

Montague, Richard (1963), Syntactical Treatments of Modality, with Corollaries on Reflexion Principles and Finite Axiomatizability, *Acta Philosophica Fennica* 16:153-167. Reprinted in (Montague 1974).

Montague, Richard (1974), *Formal Philosophy*, Yale University Press

Morgan, Charles G. (1976), Methods for Automated Theorem Proving in Nonclassical Logics, *IEEE Transactions on Computers*, vol. C-25, No. 8, August 1976

Quine, W.V.O. (1956), Quantifiers and Propositional Attitudes, *Journal of Philosophy*, 53. Reprinted in (Linsky 1971).

Quine, W.V.O. (1961), *From a Logical Point of View*, Harper and Row.

Yap, Chee K. (1977), *A Semantical Analysis of Intensional Logics*, Research Report, IBM Thomas J. Watson Research Center, Yorktown Heights, New York. RC 6893 (•29538) 12/13/77, 47 pp.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: 14 STAN-CS-79-724, AIM-325	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): 6 First Order Theories of Individual Concepts and Propositions.		5. TYPE OF REPORT & PERIOD COVERED: 9 technical rept.
		6. PERFORMING ORG. REPORT NUMBER: AIM-325
7. AUTHOR(s): John McCarthy 10		8. CONTRACT OR GRANT NUMBER(s): 15 MDA903-76-C-0206, NSF MCS 78-00524
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Artificial Intelligence Laboratory, Stanford University, Stanford, California 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: ARPA Order No. 2494
11. CONTROLLING OFFICE NAME AND ADDRESS: Eugene Stubbs, ARPA/PM, 1400 Wilson Blvd., Arlington, VA 22209	11	12. REPORT DATE: March 1979 (12) 22 p.
		13. NUMBER OF PAGES: 20
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Philip Surra, ONR Representative, Durand Aeronautics Building, Rm 165, Stanford University, Stanford, CA 94305		15. SECURITY CLASS. (of this report): 15
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Releasable without limitation on dissemination

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We discuss first order theories in which *individual concepts* are admitted as mathematical objects along with the things that *reify* them. This allows very straightforward formalizations of knowledge, belief, wanting, and necessity in ordinary first order logic without modal operators. Applications are given in philosophy and in artificial intelligence.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)